 doi:10.20944/preprints202007.0557.v1

VSM-box: general-purpose interface for biocuration and knowledge representation

Steven Vercruysse[1,2,4*], John Zobolas[2], Vasundra Touré[2], Maria K. Andersen[3], Martin Kuiper[2,4]

[1] Independent Scientist, Trondheim, Norway.
[2] Systems Biology group, Department of Biology, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.
[3] Department of Circulation and Medical Imaging, NTNU, Trondheim, Norway.
[4] These authors share senior authorship. | [*] To whom correspondence should be addressed. e-mail: vercruys@alumni.ntnu.no.
Keywords: knowledge representation, curation, biocuration, semantics, systems biology, ontology, user interface, VSM.

VSM is a recently introduced method for entering and displaying any type of knowledge, in a form that is both semantically precise for computation and intuitive for human understanding. VSM is the combination of a new semantic model, and the design for a dedicated user interface to support it. Here we present the implementation of this user interface, as a sophisticated HTML-element, <vsm-box>, that can be embedded in any web-based curation app. We show how developers can use it for biocuration projects, customize it to particular end-user needs, and contribute to its growth. Vsm-box is open-source at github.com/vsmjs/vsm-box under the AGPL license, as a JavaScript (ES6) Vue.js web-component that runs in all modern web browsers. It is the capstone of the Vsmjs organization at github.com/vsmjs that groups its supporting modules. Extensive supplementary material on VSM and vsm-box is available at vsmjs.github.io.

1 Introduction

Biocurators translate published scientific findings into structured, computable form. They may work professionally in large dedicated curation projects such as Gene Ontology, UniProt, or IntAct (Gene Ontology Consortium, 2019; UniProt Consortium, 2017; Orchard *et al.*, 2014), or they may work ad-hoc in any of the many individual research projects (International Society for Biocuration, 2018). For every new curation project, a new curation environment needs to be developed, i.e. a database and a data-entry interface (or else, curators must endure a bare-metal spreadsheet); and for every new data-capturing requirement that emerges during the lifetime of the project, the database and user-interface need to be updated (or the spreadsheet becomes ever wider and less manageable).
To alleviate this, we designed a general-purpose knowledge representation and curation approach: VSM (Visual Syntax Method) (Vercruysse and Kuiper, 2020). VSM allows scientist-curators to formulate any knowledge into an intuitive, flexible sentence-like format. Each *VSM-sentence* is a statement consisting of *VSM-terms*, which are term + identifier (ID) pairs from ontologies, controlled vocabularies (CVs), or databases (e.g. UniProt). The syntax of how VSM-terms interrelate is defined with visual *VSM-connectors* based on just a few rules, regardless of how long or complex the sentence is. VSM-sentences can be built either completely de novo, or from *VSM-templates* that guide some common structure and required terms. In a template, users only need to fill term+IDs via an autocomplete selection panel. Template-sentences remain flexible, as they can be extended with additional annotation details. This enables user-oriented capture of semantically precise information, with any amount of context details, in and across Life Science domains and likely beyond.

2 Results

Here we present the implementation of VSM's user-interface: a component that is placed on web pages with the tag `<vsm-box>`. Each *vsm-box* supports the entry or display of one VSM-sentence.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 July 2020 doi:10.20944/preprints202007.0557.v1

2.1 Vsmjs project overview

Based on practical insights from our prototype (unpublished) that biocurators used for years (Tripathi *et al.*, 2016), we have now built a production-ready, reusable vsm-box. The code is modularized as Vue.js subcomponents and as *Vsmjs* packages (Fig. 1A). These are meticulously covered by automated tests, making them robust for both end-users and co-developers. The GPL-license (cf. Linux, MongoDB) secures that any extensions made to its core remain open-source. Extensive documentation exists online. Here we summarize how developers can use and configure vsm-box and Vsmjs modules to meet particular end-user needs.

2.2 Basic use for web developers

The vsm-box code is bundled in one file, to be loaded with a `<script>`-tag (Fig. 1B). It can be installed via npm, or served directly from unpkg to get the latest version in a semver range (e.g. `@^1.0.0` is the latest compatible to v1). Vsm-box has a standalone form that bundles the Vue.js framework; a slim form for web apps that provide Vue as a module shared with other components; and its .vue-form for Vue apps.

Each vsm-box should be linked to one or more *vsm-dictionaries* that give access to the ontologies and CVs that curators need (Fig. 1B). As vocabulary data is typically served by dissimilar online APIs, we designed a unifying interface for requests: the vsm-dictionary parent-class. Any subclass *vsm-dictionary-xyz* can be implemented to translate between specification-compliant requests and some API (see §2.4).

One can place a VSM-sentence or -template in a vsm-box via `initialValue` (Fig. 1B). This JS-Object holds VSM-term and -connector data as specified online. Note that: terms' `str` label accepts Unicode; `style` stylizes parts of `str` (e.g. superscript); `classID` is a CV concept's ID (e.g. URI); and `instID` represents an instance of that concept, in this sentence. InstID is `null` while entering VSM-terms, but could be filled by a curation system when saving the sentence (making terms referable from next sentences). Finally, listeners can be notified whenever the vsm-box's content is changed (Fig. 1B).

2.3 Knowledge display and entry for biologists

Users can view information in a vsm-box, or enter VSM-terms and -connectors as proposed in (Vercruysse and Kuiper, 2020) (animations online). VSM-sentences and -templates can be made to resemble natural language, as VSM-terms can show either an official term, official synonym, or unofficial synonym, coupled to an official ID; e.g. Fig. 1C uses the preposition "in" instead of the verb "has location", and verbifies the Gene Ontology term "core promoter binding" to "binds…" (Vercruysse and Kuiper, 2012). Empty fields can be linked to specific CVs from which autocomplete suggests term+IDs, and to one-click choices that appear before users type text. Templates remain extensible, so curators can still add unanticipated context information found in literature. Mouse-hovered VSM-terms (double-clicked if empty) show a Popup component with ID, metadata, linked CVs, etc. Other features include: draggable terms; and auto-sorting connectors for natural-looking stacking order.

2.4 Vsm-dictionaries primer

vsm-dictionary is a scaffold for any subclass-package that implements interaction with a term-provider's API. As some resources (like BioPortal (Noy *et al.*, 2009)) serve data from multiple CVs, a vsm-dictionary is divided into virtual *sub-dictionaries* (each with its own `dictID`). A vsm-dictionary supports autocomplete-based search: when given a partial term-string, it fetches term+ID+metadata units from the API. Metadata may include synonyms, or a *z-object* with dictionary-specific data (e.g.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 July 2020 doi:10.20944/preprints202007.0557.v1

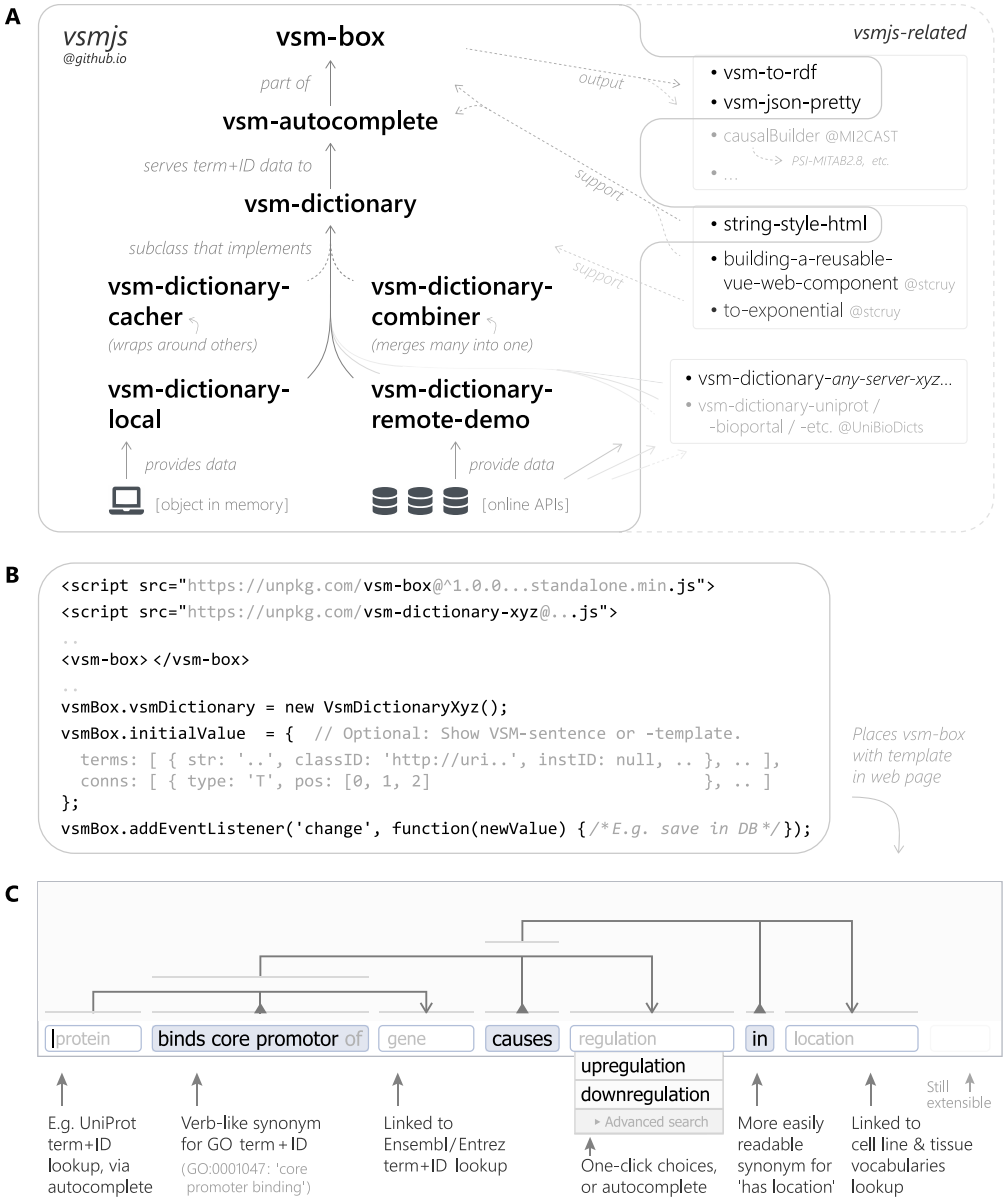

Fig. 1. Code organization and use. (**A**) Vsm-box is the apex of the Vsmjs project. It accesses term+ID data through vsm-dictionary-* subclasses that interact with vocabulary-data providers' APIs. For biological data, see UniBioDicts (Zobolas *et al.*, 2020). (**B**) Minimal code for showing one vsm-box, linked to one vsm-dictionary, filled with a template, and notifying of change. (**C**) A vsm-box user-interface element that holds a template; plus functionality descriptions. For a vsm-box application, see CausalBuilder, which also produces MI2CAST and PSI-MITAB2.8 compliant data (Touré *et al.*, 2020a, 2020b; Perfetto *et al.*, 2019).

taxon). We recommend that newly created vsm-dictionaries serve globally unique URIs for IDs/dictIDs (even if their API would not), to make curated VSM-sentences shareable, and CVs identifiable.
Results must be paginated, so that vsm-dictionaries can support also more advanced term lookup; e.g. in a future, larger dialog window. This dialog may eventually allow users to create and manage their own new terms. As PubDictionaries' API supports this (Kim *et al.*, 2019), vsm-dictionary already defines a full Create, Read, Update, Delete (CRUD) spec (only Read is mandatory).
vsm-dictionary-local is a full CRUD-implementation that serves term-data from a given object in local memory. This is instrumental when developing vsm-box features or tests, as it avoids the need for online term-providers. It is also useful as a means to serve missing terms (not yet in CVs) that curators

need. vsm-dictionary-remote-demo demonstrates interaction with an online REST API. This inspired the UniBioDicts suite of biology-related vsm-dictionaries (Zobolas *et al.*, 2020).
vsm-dictionary-combiner joins multiple vsm-dictionaries into a single one, which would be given to a vsm-box needing terms from all. vsm-dictionary-cacher wraps any dictionary in an advanced cache layer. vsm-autocomplete is embedded into vsm-box; yet can also function as a standalone web-component for CV-based autocomplete.

2.5 Extensive customization options

Vsm-box appearance is customizable via CSS (example online) (and self-adjusts to font-size), and via the `sizes` object (e.g.: horizontal connector-spacing).
Autocomplete-panel items' content is fully configurable (term, description, free-form part, etc.). One could for instance customize items based on associated dictID, add taxon icons, or add extra information (e.g. in association with a vsm-dictionary's custom `z`-object). VSM-term and Popup content is similarly adjustable.
Autocomplete-panels can show a last item with some special function: in Fig. 1C, selecting it would give control to plugin-code for "Advanced Search" of terms (e.g. in a paginated dialog). Alternatively, a "Create {typed-string}" last item creates VSM-terms with `null` as classID. This supports biologists' frequent wish to create annotation terms ad-hoc (since CVs are often incomplete) without interruption, for later review. It also enables using a vsm-box without linked dictionaries, e.g. for tryout, demos, or template drafting.

3 Conclusion

Biocuration is a process of knowledge extraction and formalization, and sharing through a database. This process should be standardized and follow curation guidelines, yet should also be intuitive, have safeguards against errors, have ample support to easily find terms and IDs, and use tools that are easy to deploy and reconfigure. We developed vsm-box to support such knowledge curation in biology, or any other domain for that matter; and we make it available as high-quality software to facilitate its adoption and further development. Vsm-box and the Vsmjs GitHub organization enable the scientific community to build new VSM-based projects, and to contribute to this technology.

Acknowledgements We thank scientists at NTNU, EBI, ISB, GRECO, and UC Berkeley/LNBL for feedback; especially A. Lægreid, M. L. Acencio, S. Tripathi, L. Thommesen, and MK who used the earlier prototype. SV sincerely thanks MK for grant co-writing and moral support during years of unpaid R&D; and JZ and VT as early-adopter addon/app developers.

Funding This work was supported by Steven Vercruysse [2011-'16,'20]; the Research Council of Norway [247727/O70]; and Patreon [stevencruy].

References

Gene Ontology Consortium (2019) The Gene Ontology Resource: 20 years and still GOing strong. *Nucleic Acids Res.*, **47**, D330–D338.
International Society for Biocuration (2018) Biocuration: Distilling data into knowledge. *PLoS Biol.*, **16**, e2002846.
Kim,J.-D. *et al.* (2019) Open Agile text mining for bioinformatics: the PubAnnotation ecosystem. *Bioinformatics*, **35**, 4372–4380.
Noy,N.F. *et al.* (2009) BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Res.*, **37**, W170–W173.
Orchard,S. *et al.* (2014) The MIntAct project--IntAct as a common curation platform for 11 molecular interaction databases. *Nucleic Acids Res.*, **42**, D358–D363.
Perfetto,L. *et al.* (2019) CausalTAB: the PSI-MITAB 2.8 updated format for signalling data representation and dissemination. *Bioinformatics*, **35**, 3779–3785.
Touré,V. *et al.* (2020) CausalBuilder: bringing the MI2CAST causal interaction annotation standard to the curator. Preprint at https://doi.org/10.20944/preprints202007…-*soon*.
Touré,V. *et al.* (2020) The Minimum Information about a Molecular Interaction Causal Statement (MI2CAST), *Bioinformatics*, , btaa622.
Tripathi,S. *et al.* (2016) Gene regulation knowledge commons: community action takes care of DNA binding transcription factors. *Database*, **2016**, baw088.
UniProt Consortium (2017) UniProt: the universal protein knowledgebase. *Nucleic Acids Res*, **45**, D158–D169.
Vercruysse,S. and Kuiper,M. (2012) Jointly creating digital abstracts: dealing with synonymy and polysemy. *BMC Res. Notes*, **5**, 601.
Vercruysse,S. and Kuiper,M. (2020) Intuitive Representation of Computable Knowledge. Preprint at https://doi.org/10.20944/preprints202007.0486.v2.
Zobolas,J. *et al.* (2020) UniBioDicts: Unified access to Biological Dictionaries. Preprint at https://doi.org/10.20944/preprints202007…-*soon*.